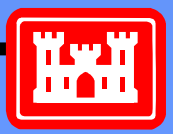
Improving Performance Using Cpusets on the SGI Origin 3800

Jeff Hensley

ERDC MSRC

January 9, 2002



User complaints
“Inconsistent performance on ruby”

- Some users were reporting inconsistencies in the time to run their codes on ruby (512 processor SGI O3K)
  - similar or identical jobs could take significantly different amounts of time
  - difficult to analyze or reproduce (results appeared to be random)
- “Common knowledge” that on single image SGI O3Ks with large numbers of processors that such behavior occurs.

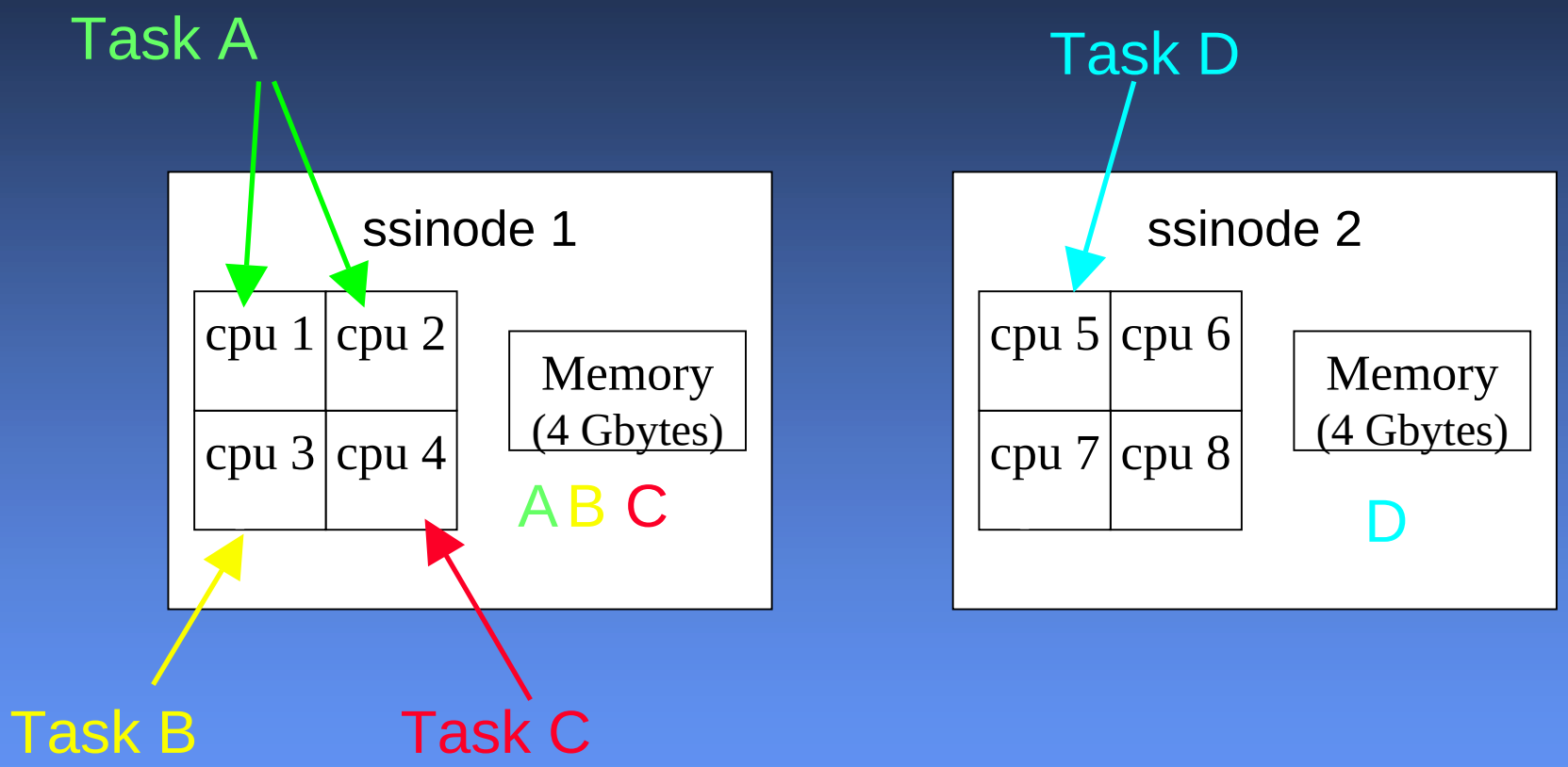
What's the cause of this behavior?

- Competition with system tasks
  - On a fully loaded machine, system tasks must use the same resources as user jobs
- Migrating processes
  - Processes are not bound to specific processors
  - Operating system can move the processes around
  - *Data* does not migrate → a process may be accessing memory located "far away"

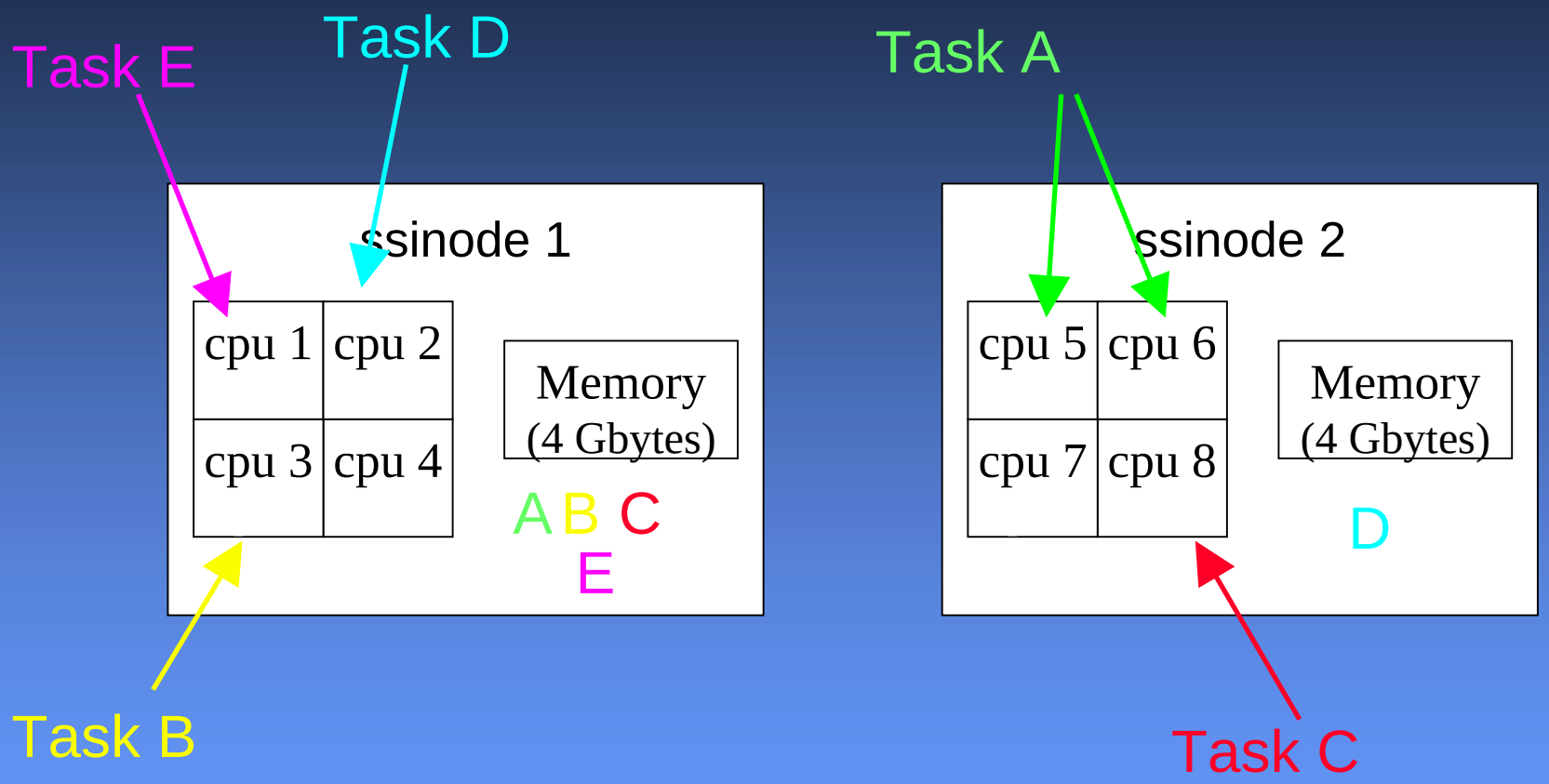
What's the cause of this behavior?

- The "SGI Shuffle"
  - Tasks are not "stuck" to a processor; the operating system moves tasks around between processors during execution
  - Data does ***not*** migrate

“SGI Shuffle”


Task A
Task D
ssinode 1
ssinode 2
cpu 1
cpu 2
cpu 3
cpu 4
cpu 5
cpu 6
cpu 7
cpu 8
Memory
(4 Gbytes)
Memory
(4 Gbytes)
A B C
D
Task B
Task C

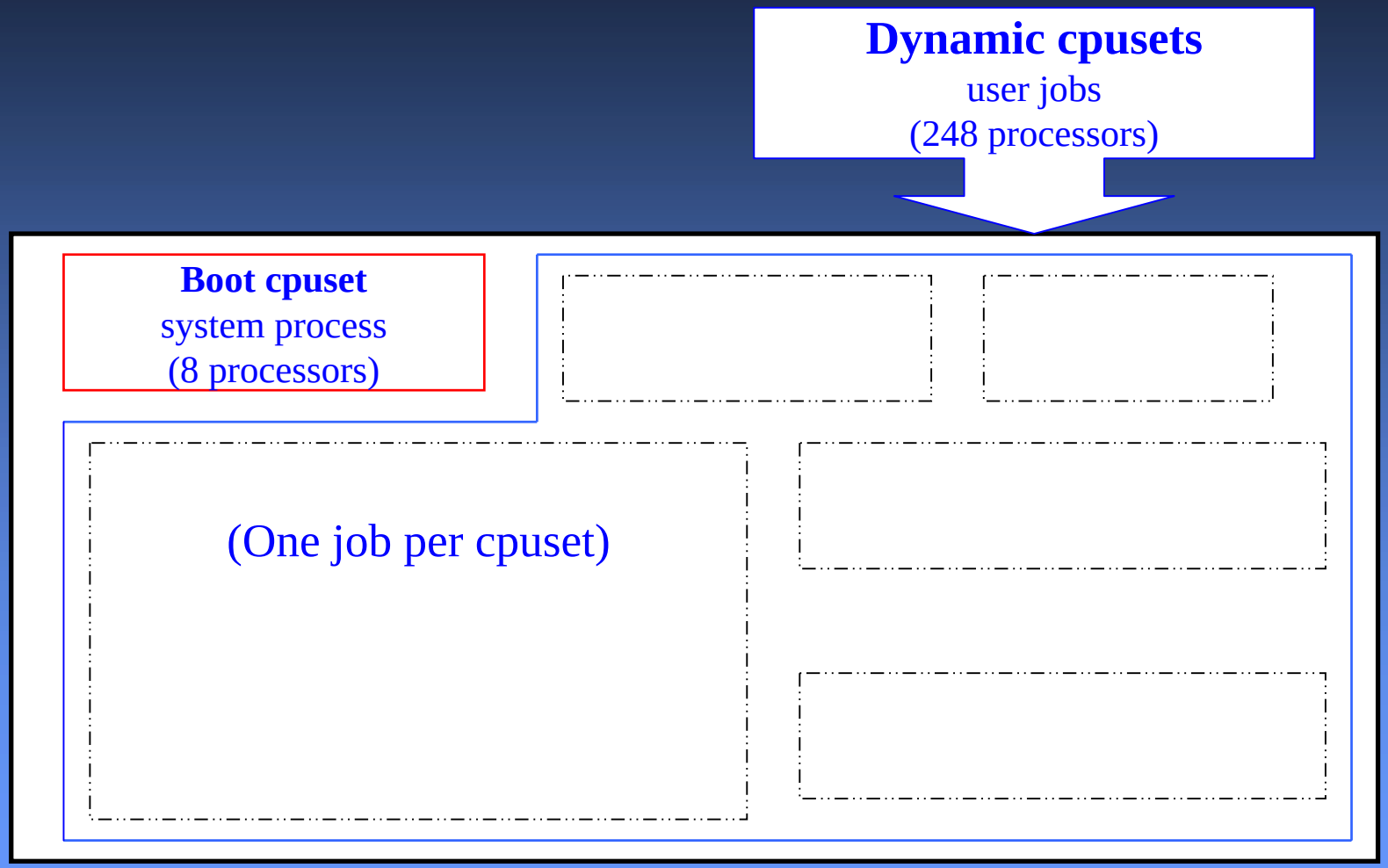
“SGI Shuffle”


Task E
Task D
Task A
ssinode 1
cpu 1
cpu 2
cpu 3
cpu 4
Memory
(4 Gbytes)
A B C
E
ssinode 2
cpu 5
cpu 6
cpu 7
cpu 8
Memory
(4 Gbytes)
D
Task B
Task C

Enter cpusets

- Cpusets are a logical partitioning of processors and/or memory
- “boot cpuset” created at system startup --- handles system tasks and daemons
- When PBS starts a job, a cpuset is created and the job runs within that cpuset
- Other tasks are not allowed access to the resources of the cpuset

Cpuset partitioning -- schematic



CPUSETS

- Most large single image Origin systems are configured to use cpusets
- “Common” knowledge that cpusets improve performance
- However, it is difficult to find the results of any tests that quantify the problem or measure its magnitude

Experimental Results

- SGI provided access to a 256-processor O3K
- Experiments included throughput tests to measure the performance of the system both with and without cpusets implemented
- The results of the tests were striking (and surprising)

Throughput Test

- Constructed a throughput test consisting of runs of 6 different codes
  - All codes were MPI codes
- Various input decks and processor counts
  - 15 distinct runs with multiple copies of many

Throughput Test

- For the throughput test, the jobs were submitted via PBS. The machine was heavily loaded during the test.
  - The throughput test was run 3 times with the machine configured without cpusets and 3 times with the machine configured with cpusets.

Throughput Test

	CPU Time (hours) for entire test	
Test number	Without cpusets	With cpusets
1	1116	747.41
2	1200	746.85
3	1298	747.14

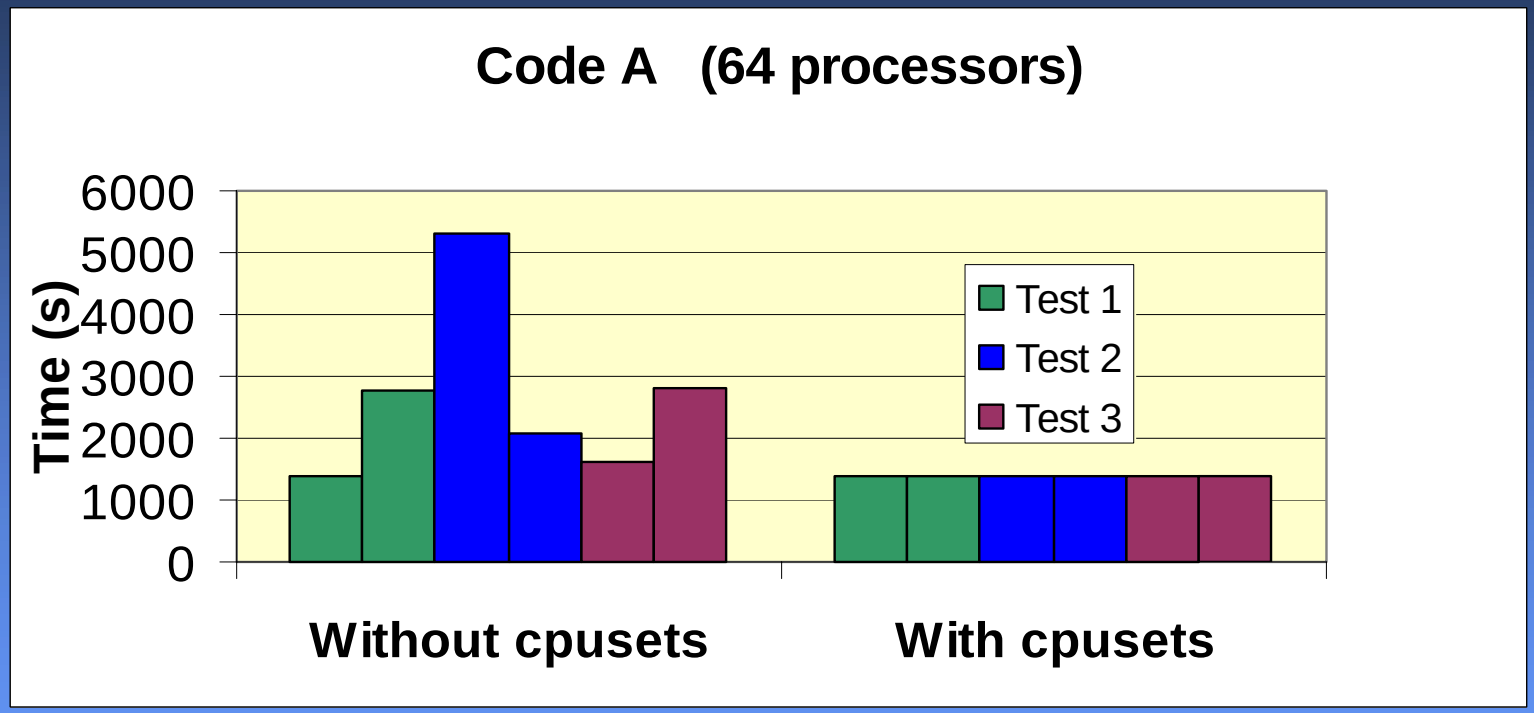
Throughput Test

- For the 3 tests, using cpusets saved approximately
  - 33%
  - 38%
  - 42%

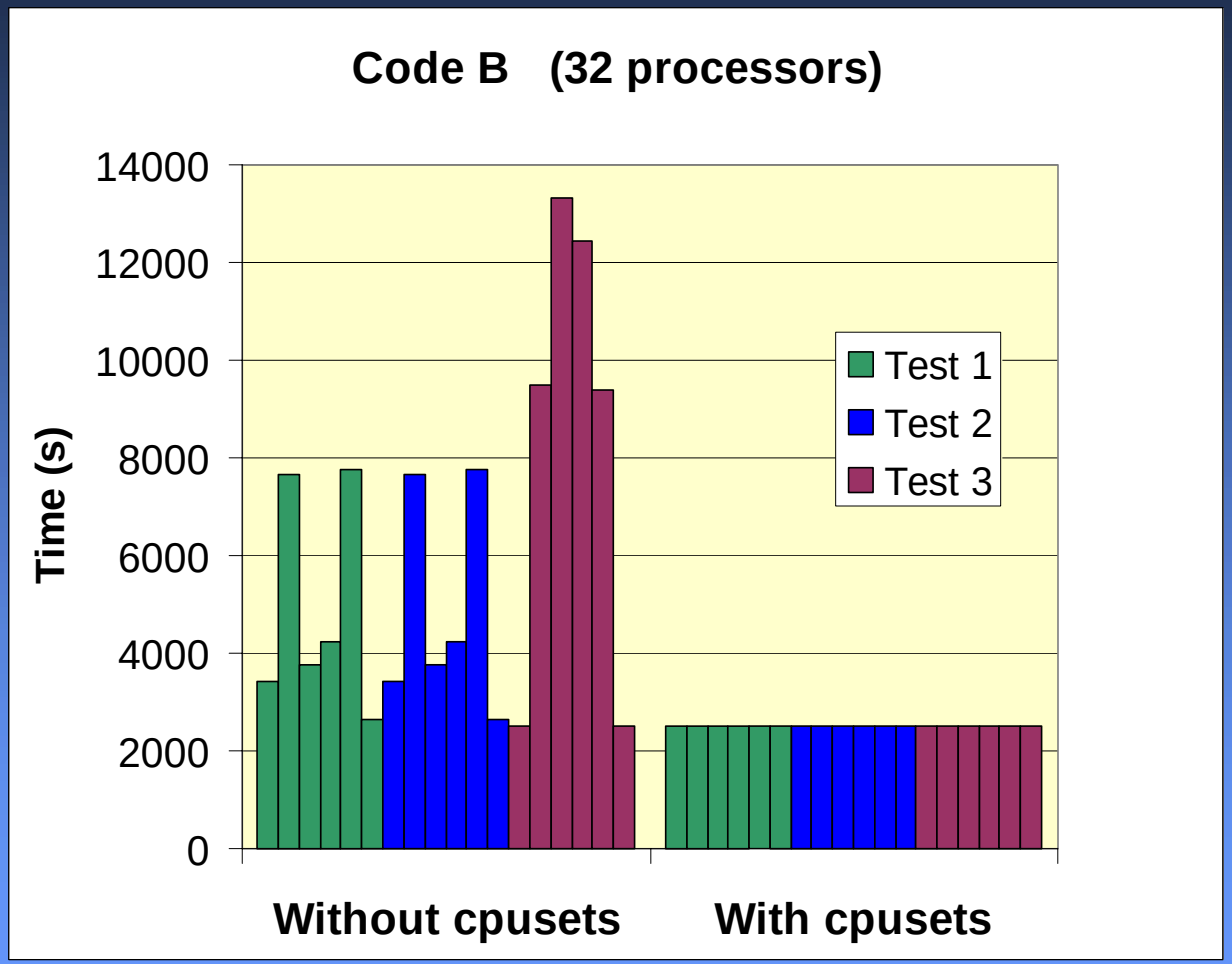
  cpu time compared without using cpusets

- Note that there is very little variation in the times for the 3 tests using cpusets

Code A


Code A (64 processors)
Time (s)
6000
5000
4000
3000
2000
1000
0
Test 1
Test 2
Test 3
Without cpusets
With cpusets

Code B


Code B (32 processors)
14000
12000
10000
8000
6000
4000
2000
0
Time (s)
Test 1
Test 2
Test 3
Without cpusets
With cpusets

Additional Test

- Question: should we use all the processors?
- Used two codes that scale well and submitted jobs that used all or most of the processors on the machine

Using all the processors?

CODE 1				
# of processors	**with cpusets**	**without cpusets**		
		1	2	3
248	616	627	627	630
256	n/a	621	630	625

CODE 2				
# of processors	**with cpusets**	**without cpusets**		
		1	2	3
248	1676	1682	1684	1704
256	n/a	1730	1719	1725

Time in seconds for job run

Experiment 2 - 512 cpus

- Brief access to a 512-processor O3K
- Constructed a simpler throughput test consisting of runs of 5 different user codes (2 of which were serial codes)
- Various input decks and processor counts
  - 14 distinct runs with multiple copies of many

Throughput test on 512 processors

- Only ran throughput test once each with cpusets and without cpusets
- Total cpu time
  - 961 hours --- without cpusets
  - 744 hours --- with cpusets
- Saved 23% cpu time by using cpusets

Throughput test on 512 processors

- Why not the same size improvement as observed in Experiment 1?
  - Didn’t load the machine up as well
  - Different mix of jobs
  - Shorter wall clock time – jobs at the end were actually running “almost dedicated”
- Did include serial jobs in this test.
  - using cpusets saved 11% on all of the serial tasks.

Conclusions

- cpusets are a good idea!
- On a heavily loaded large scale SGI Origin there will be a significant increase in average job throughput.
- Improved performance will be seen on both parallel and serial codes (however, there will be some codes which show little difference).

Comments

- One thing to be remembered: HPC systems are still “community” assets
  What one user does on the machine can effect the performance of another application
  - Competition for processors/memory
  - I/O

- We have a duty to be “responsible” users